COP 3330
Homework 5

Out: 3/19/12 (Monday)
Due: 3/30/12 (Friday) at 11:55 PM WebCourses time
Late submissions accepted until 4/1/12 (Sunday) at 11:55 PM

Objective

1. To familiarize you with creating basic GUIs using Swing.
2. To get you accustomed to using libraries based on their documentation.

Code

You may discuss high-level ideas of how to solve these problems with your classmates, but do not discuss actual source code (except for boiler-plate code such as opening a file for reading, reading user input, etc.). All source code should be yours and yours alone. **Do not:**

- Share your source code with anyone
- Ask anyone to share his/her source code with you
- Ask/pay a programming forum/community to write your homework for you
- Pair program
- Do anything else that you know is deceptive, dishonest, or misleading in any way

Problem: Fun With Fractals!

In this homework assignment, you will write a program to draw a well-known fractal – the *Sierpinski Triangle*. Fractals are very interesting shapes that are usually *self-similar*, i.e., if you zoom in on a small piece of the fractal, it looks the same as the whole thing. And if you zoom in on a small piece of that, it looks the same as the whole thing, and so on, *ad infinitum*.

The Sierpinski Triangle (see Wikipedia for a detailed description with pictures) can be described as follows. It consists of a triangle, inside of which are three smaller copies of itself. Each of these smaller triangles contains three even smaller triangles, and so on, forever. However, while it theoretically goes on forever, when it comes to drawing it in practice, it is sufficient to just draw it up to a specified depth. Beyond that, the triangles are so small that no one can tell the difference anyway. Besides, infinite computations are generally considered a bit wasteful.

The notion of *levels* is quite natural to fractals, and we can describe the triangle in terms of the related concept of *depth*. We'll number levels inside out, so if we draw 4 levels of the fractal, the outermost triangle is at depth 4, its three immediate subtriangles are at depth 3, each of their immediate subtriangles are at depth 2, etc.

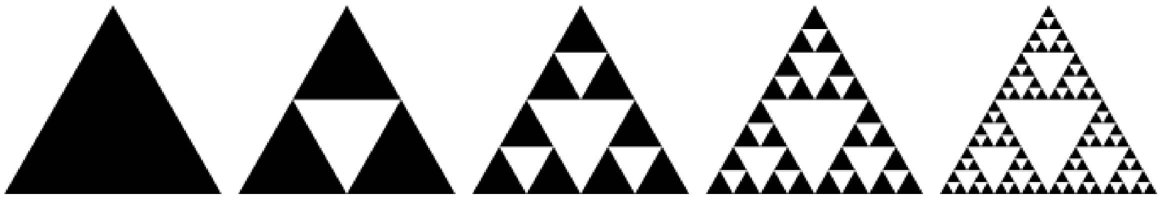


Figure 1: Sierpinski Triangles with maximum depths 1 through 5

In the figure, you can see the fractal drawn up to a set number of levels. (The colors are done a little differently from our approach, so just pay attention to the general shape of things.)

To make the fractal look really interesting, we'll color things according to the following rule. All triangles at a particular depth will have the same color. So all level 4 triangles are red, level 3 triangles are blue, level 2 triangles are green, etc. See Illustration 2 for an example of this color scheme.

To make things simple for you, we will supply you with a recursive function that will draw the triangles, with only the code for painting triangles left out. (This is what you will add.) In order to figure out how to place the triangles, you will need to know some general mathematical expressions for the locations of the vertices. We will explain this here.

Consider the following diagram of a depth-2 Sierpinski triangle. Note that we use the standard graphics coordinate system, where the origin is at the top left, and the positive y-axis comes downwards instead of upwards.

To simplify the math, we will think in terms of the *bounding square*s of the triangles. These won't be drawn by your program, but it is simpler to reformulate the problem in terms of drawing triangles inside imaginary squares. The picture shows how to draw a Sierpinski triangle inside a bounding square with upper-left corner *(x, y)* and side-length *S*.

The big triangle is straightforward – it's base extends across the base of the square, and the apex is at the midpoint of the top side. The coordinates are given in the figure.

For the subtriangles, we think of terms of the three smaller bounding squares, which each have side-length *S/2*. We follow the same rule as above, only with the smaller squares. Notice that the big triangle is yellow, and the three smaller ones are red. (The upside-down triangle in the middle is a hole, through which we can see the color of the big triangle.)

This idea is easily extended to higher depths – simply repeat the process for each bounding square, until you reach the desired depth.

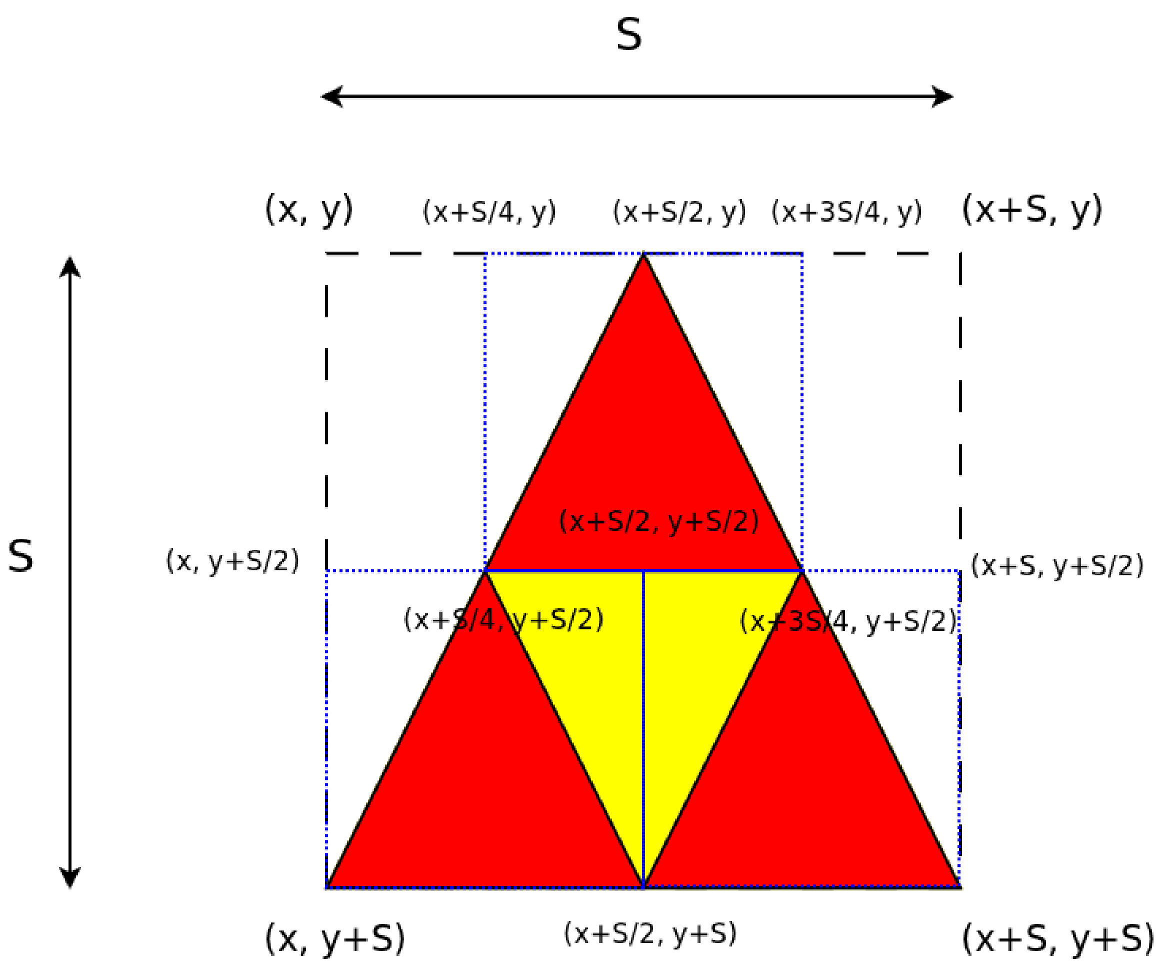


Figure 2: Depth-2 Sierpinski triangle in bounding box with top left corner (x, y) and side-length S

<u>Code</u>

Here is a recursive function that draws the triangle up to depth *d*, starting with a bounding square at *(x, y)*, with side-length *S*.

```
/*
 * This will eventually be called somewhere as:
 * draw(d, 0, 0, 512);
 *
 * This just means that the big bounding box is at the upper-left corner (0, 0) of your Canvas, since we
 * want to fill it out completely.
 */

void draw(int d, int x, int y, int S) {
      // Done drawing, stop.
      if(d == 0) return;

      // Otherwise, draw big triangle at this level, between the points
      // shown in the figure. You can use the fillPolygon() method of
      // the Graphics object of your Canvas. Make sure you get the color
      // right!

      /*<INSERT YOUR CODE HERE>*/

      // Draw the subtriangles. The self-similarity of fractals means
      // that they are themselves Sierpinski triangles of depth d-1.
      draw(d-1, x+S/4, y, S/2);
      draw(d-1, x, y+S/2, S/2);
      draw(d-1, x+S/2, y+S/2, S/2);
}
```

Details

Your window will contain the following components:

1. A canvas, on which the drawing will be done. Make sure your canvas is square (not rectangular) and has dimensions 512x512. These are powers of 2, so all the divisions by 2 and 4 will come out even.
2. A text field (with a descriptive label) that specifies the depth to which the triangle should be drawn. It should pop up an error dialog if the depth entered is greater than 10.
3. Five combo boxes, from which we can select the colors for each level. If there are too few colors (maybe you're drawing it up to depth 10), your program should simply cycle back to the first one.
4. A check box for randomizing colors. When this is selected, colors should be chosen randomly (though all triangles at the same level should have the same color!) and the combo boxes should be disabled.
5. A draw button, that draws the triangle on the canvas, with the chosen colors and depth. (This may cause the text field to pop up the error dialog mentioned in 2.)

See the screenshot for an example of how it should look.

Layouts

The best way to lay things out properly is to use JPanels to group together related components, and use an appropriate layout manager for each JPanel as necessary. Here's a possible grouping.

1) Canvas
2) Text Field and Label.
3) The five combo boxes and the check box.
4) The draw button. (Or you could just put this in 3 as well)

That's four JPanels, each of which contains one group of components. You can also put a bunch of JPanels into another JPanel, so JPanels 2-4 could go into one, for the logical grouping 'Right side of the window'.

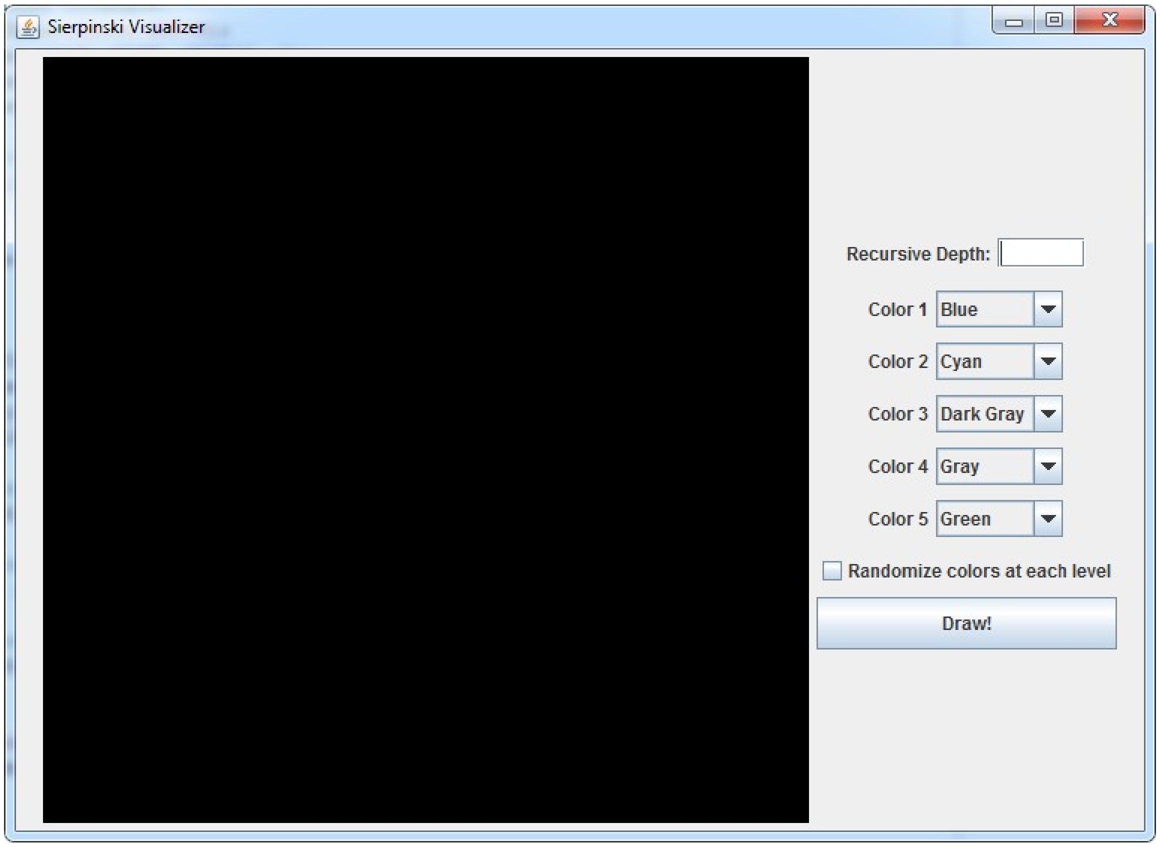
Here is a good way to do layouts:

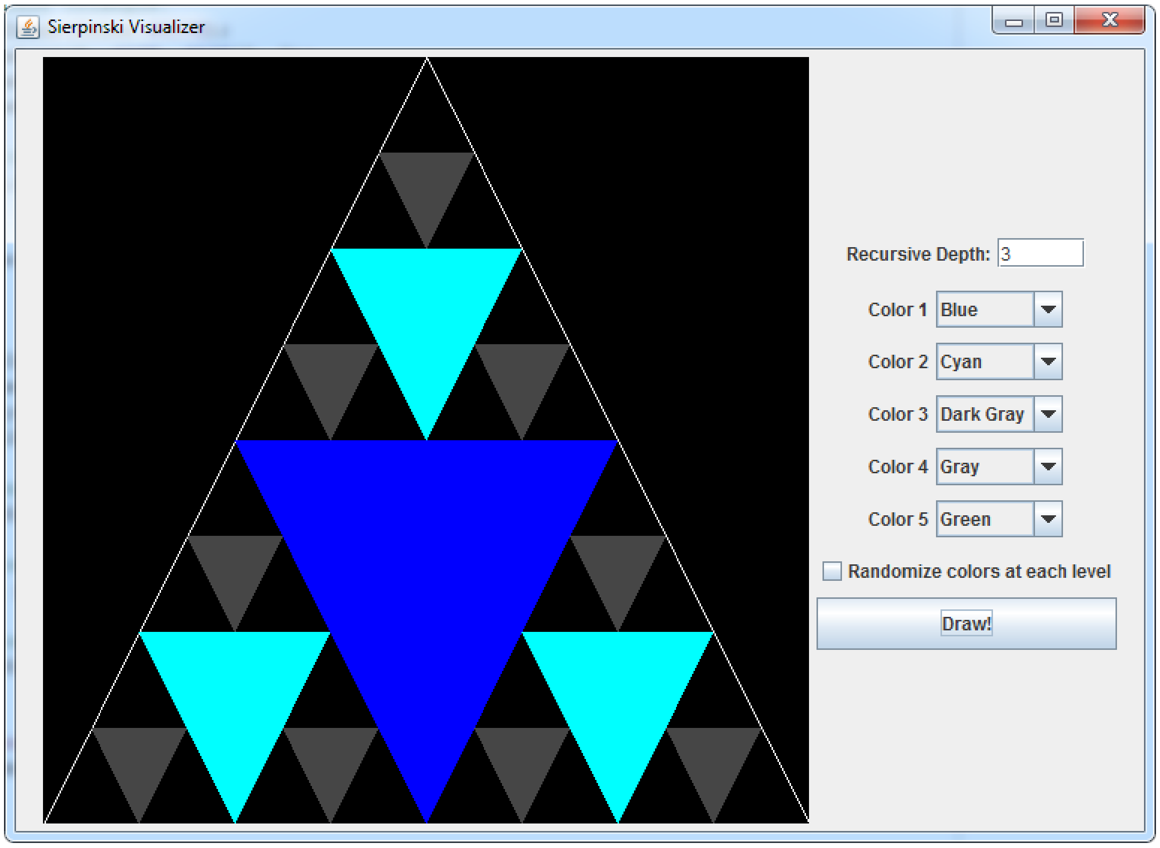
1) 1x2 GridLayout, with panel 1 on the left and the superpanel 2-4 on the right.
2) FlowLayout for panel 2.
3) BoxLayout for panel 3
4) FlowLayout for panel 4.

Alternatively, you can spend some time wrapping your brain around the mysteries of GroupLayout and do everything that way. It's a bit confusing until you get it, but very powerful when you do.

Screenshots

Here are some pictures of what the general layout of the components should be.





Recursive Depth: 3

Color 1 Blue

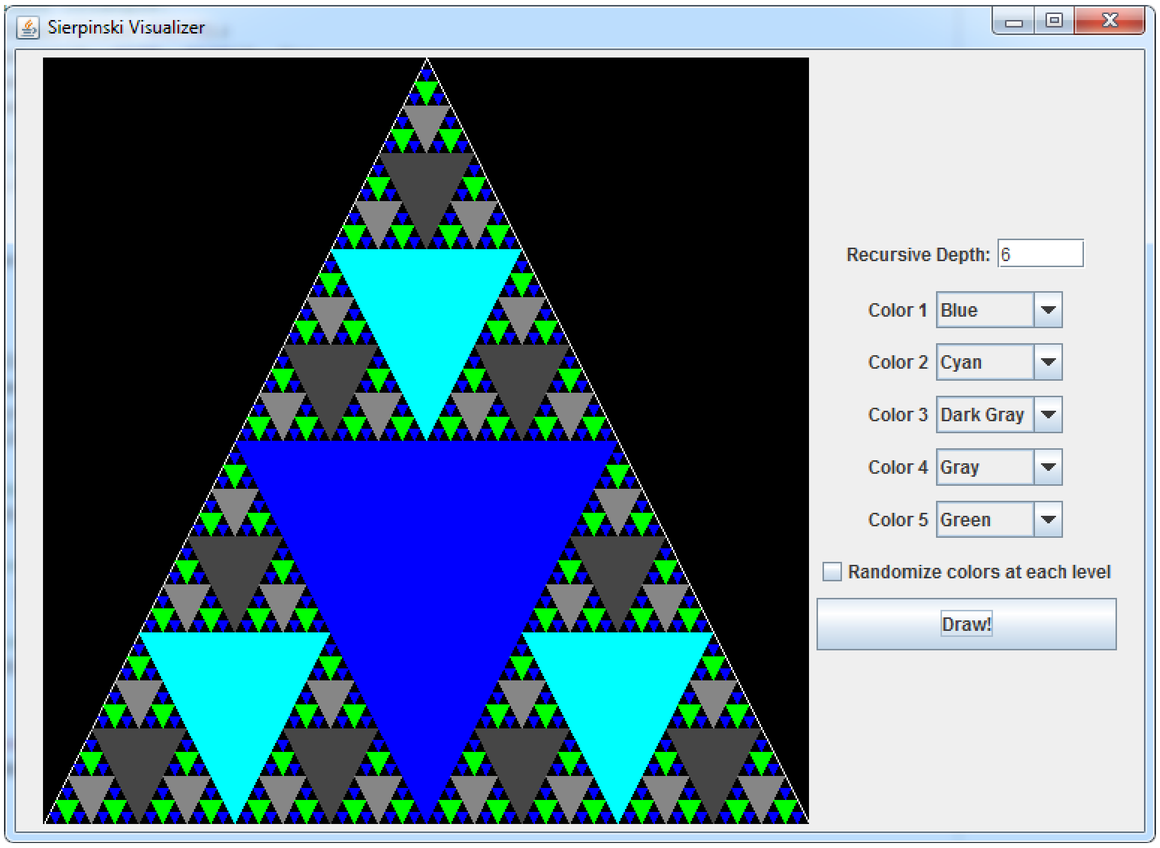
Color 2 Cyan

Color 3 Dark Gray

Color 4 Gray

Color 5 Green

Randomize colors at each level [ ]



Recursive Depth: 6

Color 1 Blue

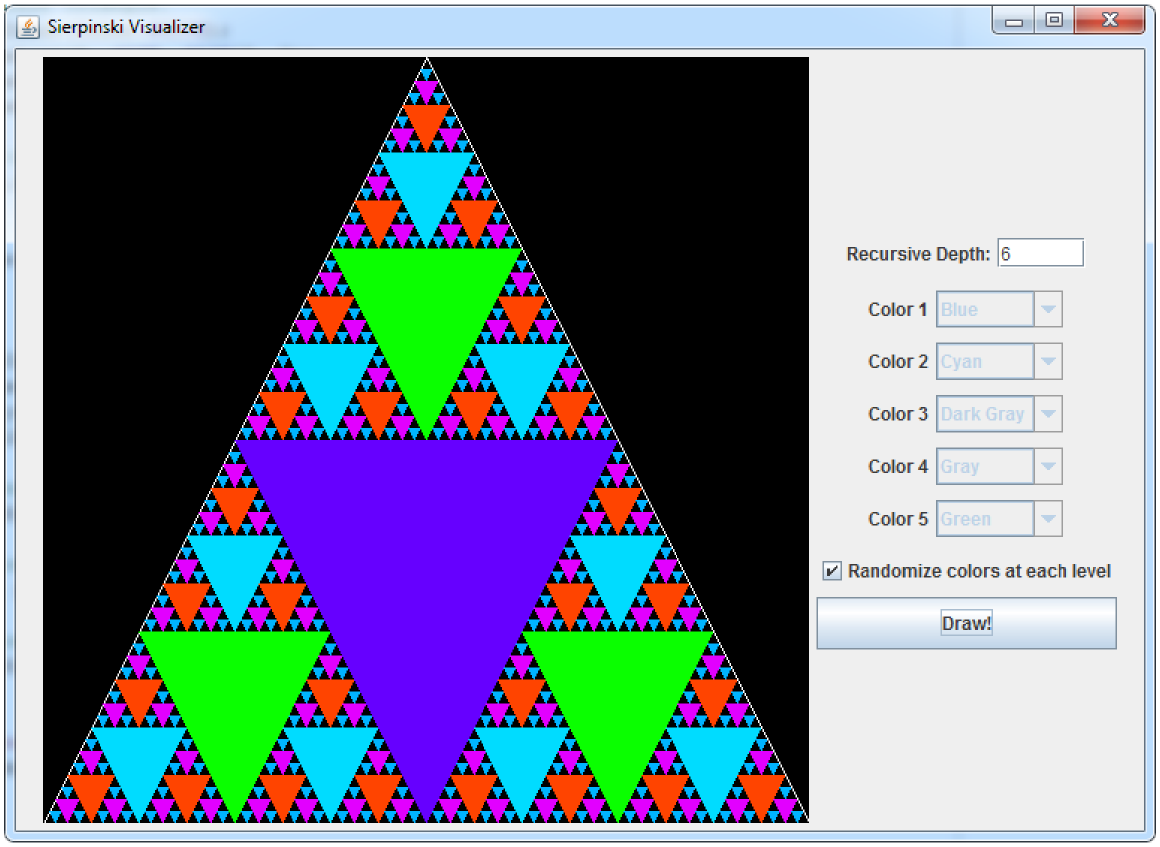
Color 2 Cyan

Color 3 Dark Gray

Color 4 Gray

Color 5 Green

Randomize colors at each level [ ]



Recursive Depth: 6
Color 1 Blue
Color 2 Cyan
Color 3 Dark Gray
Color 4 Gray
Color 5 Green
☑ Randomize colors at each level

Input/Output

It's fairly obvious when you've got this working right, but you can use Figure 1 as a rough guideline to make sure the first few depth values are working right.

Deliverables

Submit the source files over Webcourses as an attachment. **In particular, *do not* submit any .class files!!! This will result in 0 credit for the assignment.** You must send your source files as an attachment using the "Add Attachments" button. Assignments that are typed into the submission box will **not** be accepted.

Restrictions

Your program must compile using Java 6.0 or later on the command prompt. It's okay to develop your program using the IDE of your choice, but the TAs will use the command prompt to compile your code and run the programs. Your code should include a header comment with the following information:

Your name
Course number, section number
Description of the code

You **WILL** lose credit if this information is not found in the beginning of each of your programs. You should also include **inline comments** to describe different parts of your program where appropriate.

Execution and Grading Instructions

1. Download the source *.java* files and place in a folder.
2. Check the source code of each program to make sure it contains header comments, inline comments and reasonable use of variable names.
3. Run the program and test to see if all the required functionality is present.
4. If all required functionality is present and works correctly, give full credit for execution. Otherwise, give partial credit based on the situation.